Docket No.: DE920010077US1

Inventor: Hanno Ulrich
For: A FEEDBACK PROCESS TO MODEL CP UTILIZATION MULTI-PROCESSOR PERFORMANCE MODELS

APPLICATION FOR UNITED STATES

LETTERS PATENT

"Express Mail" Mailing Label No.ET251817831US
Date of Deposit: June 23, 2003

I hereby certify that this paper is being deposited with the United States Postal Service as "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to: Box Patent Application, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Name: Susan L. Nelson

Signature: Susan L. Nelson

INTERNATIONAL BUSINESS MACHINES CORPORATION

A FEEDBACK PROCESS TO MODEL CP UTILIZATION MULTI-PROCESSOR PERFORMANCE MODELS

BACKGROUND OF THE INVENTION:

[0001] The present invention relates to a method and system implementing feedback algorithms for controlling a given simulation model. In particular, it relates to a computer-program-based method and system for providing a feedback control for a given set of control quantities of a simulation model, comprising a plurality of iterated simulation runs in which a single simulation run consumes a considerable amount of time.

[0002] The present invention will be next defined from prior art on the specific field of computer system modeling. In particular, the invention will be illustrated by an exemplary application to a large scale multi-processor (MP) performance model with special focus on storage hierarchy (SH) performance.

[0003] In the inventive context the term *model* refers to a time-dependent simulation model in contrast to an analytical spread sheet model: Simulation models are true cycle-by-cycle models of computer systems. In particular, queuing delays and resource utilizations are a result of simulated data transfer. Analytical models instead use mathematical formula to calculate queuing delays from given rates and utilizations.

[0004] In large-scale compute environments, it is generally recommended that central processor (CP) utilization should not exceed 90%. Otherwise, excessive queuing delays for shared system resources can severely impact system performance. Therefore,

performance benchmarks used to test real hardware are also usually performed at the same CP utilization of 90%.

[0005] Consequently, simulation models of real hardware must be able to accurately simulate any given CP utilization. When a multi-processor model ignores CP utilization and hence runs at a utilization of 100% instead of 90%, the load on the storage hierarchy cooperating closely with said CPs is significantly overstated and not representative for real customer environments. Sample model runs ignoring CP utilization turned out to mispredict total system capacity by as much as 7.5%. A misprediction of that magnitude can lead to wrong and hence costly design decisions and - even worse - to a wrong positioning in the marketplace.

[0006] The difficulty to correctly model a given CP utilization results from the fact that CP busy time and hence CP utilization includes both pure instruction processing time and the time in which the CP is waiting for storage hierarchy (SH) requests to be resolved. The wait time in turn depends on the load of the SH and thus indirectly on CP utilization. As this functional relationship is unknown, prior art computer simulation models often ignore CP utilization completely and operate constantly at a utilization of 100% - with the exposures and consequences as mentioned above.

[0007] The inventive approach presented here applies both to computer system models driven by instruction traces and driven by event rates.

[0008] In models driven by instruction traces, the simulated CPs read, interpret and execute instruction sequences. In this case, CP utilization can be simulated by suspending instruction

processing for a while at adequate points in the instruction stream - that is by artificially inserting periods of user think time. Event driven models instead have drivers implemented which statistically generate requests to the storage hierarchy. Typical requests are L1 cache misses. Their frequency is an input parameter to the model and CP utilization is reflected in the interarrival time of events which is the average number of processor cycles between two events: The higher the CP utilization, the more requests are issued per unit of time.

[0009] In both model types, the leverage to manipulate a CP's utilization is its relative utilization during the time in which it is not waiting for a SH request to be resolved. This time is referred to herein as the CP's entry utilization χ. Then, the total utilization μ is a workload dependent and system dependent function of χ, $\mu=\mu(\chi)$ As mentioned above, μ includes CP wait times and hence is unknown, and the underlying utilization modeling problem consists in choosing χ such that $\mu=\mu(\chi)$ converges against the utilization aimed at. This utilization quantity is referred to herein as the target utilization μ.

[0010] As mentioned before, prior art computer simulation models often ignore CP utilization completely. If, however, such prior art involves the predetermined setting of CP utilization, the most obvious approach to make μ converge against μ is the so-called regulation technique.

[0011] Said prior art regulation techniques use the following iterative approach which consists of:
a) defining a start value for the entry control quantity,
b) performing a current run of simulation with this start value, thus yielding a resulting value for the target control quantity,

c) comparing the resulting target quantity to its value aimed at,
d) defining a new starting value in dependence of the existing result, and
e) performing a next run of simulation with this new starting value.

[0012] Of course, instead of being 1-dimensional, the starting value as well as the control quantity can be a multi-dimensional vector reflecting the multiple dimensions in a simulation system.

[0013] In prior art, the algorithm to choose the entry utilization works as follows: Starting with some entry value $\chi=\chi'_1$, the simulation model runs for a while, then determines the simulated total utilization μ_1 and chooses a new value for χ to be selected for the next iteration of the simulation run: If $\mu_1<\mu$, then $\chi'_2>\chi'_1$ is chosen and vice versa. χ' denotes the instantaneous value of χ valid for the next iteration. The quality and speed of convergence depend on the step-sizes chosen for the difference between χ'_2 and χ'_1 and on the initial value chosen for χ and may even vary with the value of the target utilization μ. In particular, this heuristical method is exposed to over-correct χ and hence may lead to an undesired wide oscillation of the simulated utilization around μ.

[0014] Disadvantageously, prior art provided feedback control algorithms may even become complex, as soon as the convergence history is used to determine the starting value of χ for the next iteration of the simulation run. Thus, the drawback of prior art is that:
a) it converges too slowly, which implies long-lasting simulation runs,
b) it is exposed to an undesired oscillation around the target utilization.

BRIEF SUMMARY OF THE INVENTION:

[0015] It is thus an objective of the present invention to provide a method and system which:
a) saves computational time in computer simulations working according to the above sequence principle
b} avoids wide oscillation around the target value of utilization.

[0016] In contrast to prior art algorithms as described above, the inventive algorithm uses one closed formula which was discovered within the present invention to choose the next instantaneous value χ' for the entry control quantity χ. This value applies advantageously unchanged to any target control quantity μ, like the before-mentioned target utilization, implies excellent convergence of μ, and implicitly regards the convergence history.

[0017] Thus, according to its basic aspect, a computer-program-based method for providing a feedback control for a given set of entry and target control quantities χ and μ of a system model is disclosed, the method comprising a repetition of the following steps:

a) providing a starting value χ'_1 for each of the said entry control quantities χ in the model,

b) running the model based on said starting values and obtaining a resulting actual value for each of said target control quantities μ,

c) using the values obtained for μ to define a new start value for χ for use in a repeated modeling step,

whereby the method is characterized by comprising the following formula to calculate the respective next value of the entry control quantities:

$$x'_{n+1} = \frac{v_n}{1+\rho_n(1-v_n)} \qquad (6a)$$

where ρ_n is a suitable parameter and

$$v_n = (n+1)u - nu_n \qquad (6b)$$

χ'_n is valid for the next iteration only while μ_n and ρ_n are values measured from the beginning of the simulation.

[0018] Instead of being given directly, the starting value or starting vector χ may also be provided by derivation from other setting quantities χ is dependent from, if such alternative setting means are available in the system.

[0019] When said control quantities are CP utilizations in computer system models simulating multi-processor systems, then significant simulation time can be saved, and bring-up costs and bring-up time for new-developed computer hardware can be significantly reduced.

[0020] Therefore, the inventive method and its cost and time savings are of significant interest to any computer manufacturer building large scale multi-processor systems. Beyond the field of computer system simulations, the inventive algorithm disclosed also applies to and is beneficial to any field of technical simulations where a single simulation run consumes a considerable amount of time which is desired to be shortened.

[0021] However, its applicability is not restricted to simulation models. The inventive method may also be used to control any real production process, provided the process has the following characteristics:

first, the long term utilization of the processing machine must not exceed a certain limit (maybe in order to avoid burning out of temperature-sensitive parts),

second, every once and a while, the work piece must be taken out of the main production stream for side processing (such as for heating up) before main processing can continue,

third, the duration of the side process is not predictable, such as, when several main production lines compete for only one side processing facility, and,

finally, the main processing system may not be turned off during side processing, because the work piece must be further processed immediately when it returns to the main production line.

[0022] In that case, in order to keep the main processing system's long term utilization below the given limit, but as well near to that limit, the system must be turned off for a certain amount of time every once and a while. As a skilled person may appreciate, this situation is equivalent to the CP utilization problem in an instruction driven multi-processor simulation model

as described above. Examples in the above sense are steel and rolling mills, or in particular bio-chemical production processes.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING:

[0023] These and other objects will be apparent to one skilled in the art from the following detailed description of the invention taken in conjunction with the accompanying drawings in which:

[0024] Fig. 1 is a schematic block diagram representation of a system under observation and simulation, to which the inventive approach applies;

[0025] Fig. 2 is a schematic block diagram representation illustrating the functioning of the inventive control mechanism when applied to a plurality of transportation or processing units the throughput of which is dependent on external parameters;

[0026] Fig. 3 is a schematic block diagram representation illustrating a multi-processor performance model which is controlled and improved by the inventive method;

[0027] Fig. 4 is a schematic block diagram representation illustrating the system structure of the IBM eServer zSeries 900 server as an example for a real large scale multi-processor system which can underlie the model structure shown in Fig. 3; and

[0028] Fig. 5 is a chart showing CP utilization over runtime illustrating the efficiency of the inventive feedback algorithm when compared to prior art control methods.

DETAILED DESCRIPTION OF THE INVENTION:

[0029] The inventive method is described in more detail by way of example applied to computer system simulation models.

[0030] The following time and utilization variables are used:

[0031] Total run time τ is the sum of system idle time ι, *instruction* processing time *P* and system wait time *W*:

$$T = I + P + W. \tag{1}$$

During *P,* the system is assumed to process instructions at infinite cache speed. *W* is the total time a CP is delayed waiting for a storage hierarchy (SH) request to be resolved. Then, total CP utilization at time τ is

$$u_T = (P + W)/T, \tag{2}$$

and the "entry control quantity", i.e., the CP's entry utilization - i.e., its relative utilization during *P* - becomes

$$x_T = P/(P + I). \tag{3}$$

[0032] Straightforward arithmetic shows

$$u_T = \frac{(1+\rho_T)x_T}{1+\rho_T x_T} \tag{4}$$

[0033] Where $\rho_T = W/P$, or equivalently

$$x_T = \frac{u_T}{1+\rho_T(1-u_T)}. \qquad (5)$$

[0034] The feedback-process implemented according to this preferred embodiment adjusts χ every Δ cycles in such a way that the actual CP utilization approaches the target utilization aimed at, which is referred to as "target control quantity".

[0035] In the following formulas,
μ denotes the *target utilization* finally aimed at,
$\Delta_n = [T_{n-1}, T_n]$, with $T_n = n\Delta$ is the n-th interval of observation time,
I'_n, P'_n and W'_n denote the idle, processing and wait time within Δ_n,
I_n, P_n and W_n the respective accumulative times up to T_n,
μ'_n and χ'_n denote the instantaneous total and entry utilization within Δ_n, and
μ_n and χ_n the respective accumulative utilizations up to T_n.

[0036] Relying on formula (5), all information available at time τ_n is used to choose the next value for the entry utilization. Thus, the inventive key regulation formula reads:

$$x'_{n+1} = \frac{v_n}{1+\rho_n(1-v_n)} \qquad (6a)$$

where $\rho_n = W_n/P_n$ and

$$v_n = (n+1)u - nu_n \qquad (6b)$$

[0037] Since μ_n aims to approach μ, the simple choice

$$x_{n+1}' = \frac{u}{1+\rho_n(1-u)} \qquad (7)$$

also makes CP utilization converge against μ - slower, however, than with formula (6a, 6b).

[0038] The inventive embodiment of the feedback process is disclosed as follows:

[0039] Assuming $W_1=P_1$ or $\rho_1=1$, the iteration starts with the entry utilization $\chi'_1=\chi_1=\mu/(2-\mu)$, where μ is the target utilization.

[0040] In a rate driven model, this determines the request rate during non-wait time in $\Delta_1=[T_0,T_1]$. In a model driven by instruction traces, it determines the user think time - i.e. the length of idle periods to be inserted into the instruction stream within Δ_1.

[0041] During Δ_1, the model records the simulated wait time $W'_1=W_1$ and at time T_1, $P'_1=P_1$ and $\rho_1=W_1/P_1$ are deduced from formulas (1) $\Delta-W'_1=P'_1+I'_1$ and (3) $\chi'_1=P'_1/(P'_1+I'_1)$.

[0042] From (2), $\mu_1=(P_1+W_1)/T_1$ is calculated, and formula (6) is applied to arrive at the entry utilization χ'_2 to be used in $\Delta_2=[T_1,T_2]$.

[0043] During Δ_2, the model records the simulated wait time W'_2, and at time T_1 similarly like before at time T_1, P'_2 and hence $P_2=P_1+P'_2$, $W_2=W_1+W'_2$ and $\rho_2=W_2/P_2$ are deduced from formulas (1) $\Delta-W'_2=P'_2+I'_2$ and (3) $\chi'_2=P'_2/(P'_2+I'_2)$.

[0044] Then from (2), $\mu_2=(P_2+W_2)/T_2$ is calculated and the procedure can be iterated as often as required by any finish-criterion.

[0045] The following reasoning is added in here, in order to make the formula easier to understand:

[0046] The instantaneous version of formula (5) valid within Δ_n reads

$$x_n' = \frac{u_n'}{1+\rho_n'\left(1-u_n'\right)} \tag{8}$$

[0047] Since obviously

$$u_n' = nu_n - (n-1)u_{n-1} \tag{9}$$

(8) turns into (6), if μ_{n+1} is assumed to be sufficiently close to μ and hence ρ'_{n+1} close to ρ_n.

[0048] Further, formula (5) suggests that the accumulated entry utilization at time T_{n+1} will be close to

$$x_{n+1} = \frac{u}{1+\rho_n(1-u)} \tag{10}$$

[0049] This is equivalent to the iterative formula

$$x_{n+1} = \frac{x_n}{1+d_n(1+\rho_n x_n)} \qquad (11)$$

where $d_n=(\mu_n-\mu)/\mu$.

[0050] From (11), it follows that $\chi_{n+1}\leq\chi_n$ and hence $\mu_{n+1}\leq\mu_n$ if $\mu_n\geq\mu$ and vice versa. This guarantees that $\{\mu_n\}$ in fact converges against the target CP utilization μ or at least oscillates sufficiently close around μ.

[0051] With general reference to the figures and with special reference now to Fig. 1 the considerably large field of use for the inventive algorithm is intended to be illustrated. The actual core of the present invention is the formula (6a, 6b) given above.

[0052] This formula can be used according to the present invention in any technical application that uses a feedback mechanism to control a system of working units 10A, 10B, …, 10D which are switched in parallel and which access a common resource 14. Each of said units may produce an output of material or immaterial nature. Examples are working pieces treated in such units 10, like for example a car in the car producing industry, or an electrical current as an example for an immaterial output.

[0053] Each of said units 10 is controlled by an associated driver unit 12A, 12B, …, 12D. Without restricting the generality of the present invention, a single driver unit 12 may also control more than one of the working units 10.

[0054] Each working unit 10 is connected to said common, shared resource 14. The individual workload capacity of each unit depends on the degree up to which said common shared resource 14 is loaded.

[0055] When said common resource 14 performs poorly, the throughput of the working units 10 can be impacted severely. The inventive algorithm is intended to control the activity or load factor of each of the working units 10 in such a way that the shared resource 14 is in an operational status which is basically non-blocking. The reason, why an overload of the common shared resource 14 may result in a blocking of this system component will, of course, vary according to the applied field of use. A central feature of the system is the interdependency between the working units: The throughput of each individual working unit 10 depends on the load of the other working units 10.

[0056] Fig. 2 illustrates a more general exploitation of the inventive control mechanism. It applies to any transportation, mailing or processing units whose throughput depends on parameters outside to the unit itself. Again, the respective unit under observation is depicted with reference sign 10. It is driven by some driver unit 12. Its throughput is limited by some throughput limitation 14 which may vary from case to case.

[0057] The inventive mechanism presented is used to keep the throughput of the working unit 10 within a given tolerance band of for example +-2% around a given required throughput of e.g. 90% of the theoretically maximum throughput.

[0058] Thus, the working units' throughput is the target control quantity μ in the sense of the appended claims. The throughput is synonymously denoted with a more generalized term

of 'utilization' of the working unit 10. The load imposed by the driver unit is the working unit's entry utilization χ'.

[0059] According to the generalized approach, a utilization measurement facility 20 measures the actual utilization μ_n of the working unit 10 by means of a predetermined scanning scheme and reports the measured utilization values to a control element 22. Measurement and report frequency may vary according to any physical requirements present in the respective application case.

[0060] The control element 22 stores a predetermined value of target utilization μ associated with the working unit 10. The control element 22 processes the measured actual utilization μ_n by means of the inventive formula (6a, 6b) and calculates a new entry utilization χ'_{n+1} which the driver unit 12 uses to drive the working unit 10 until the next utilization measurement is performed. Thus, a closed loop control is implemented, involving the inventive formula.

[0061] The loop connects two subsequent observation periods by providing a new start value for the entry utilization of the working unit 10. By means of the inventive formula, the new entry utilization is calculated from the current utilization of the working unit 10, which has just been measured.

[0062] In Fig. 2, only one working unit 10, driver unit 12 and utilization measurement facility 20 are depicted. It shall be understood, however, that multiple copies of these devices may co-exist. The input from each of the respective utilization measurement facilities 20 can than be fed into one common, single control element, which performs the above-mentioned calculations and which in turn provides a new starting value to each of said driver units 12. Thus, as can be appreciated by a person skilled

in the art, any prior art closed loop feedback control can take profit from the inventive algorithm. The advantage that results depends on the actual technical area in use. When, for example, the system is a complex simulation model, the technical effect associated with the present invention is to save computation time. When, however, the technical effect produced by the present invention is to provide for a very fast converging behavior, i.e. when by virtue of the present invention, the target control quantity such as the throughput of the working unit can be controlled such that some required value is quickly reached in practice, then, any individual physical technical effect and advantage can be achieved, the nature of which varies from the applicational field, again. When, for example the watermark of a system of rivers shall be controlled by the inventive approach, it is very important to avoid peaks of the control quantity (watermark) because both, a positive and a negative excess of the tolerated tolerance band may have severe effects on the ships being present on the rivers, or for inhabitants and infrastructure of the river's embedding landscape.

[0063] Fig. 3 illustrates a preferred implementation of the inventive approach in a multi-processor performance model. As an example for a possibly underlying real computer system, Fig. 4 depicts the structure of the IBM zSeries 900 server.

[0064] In Fig. 3, the system under observation is the box with reference sign 30. It corresponds to the total of Fig. 1. Its major hardware components are a given number of processors, each including a level 1 cache (L1-cache), depicted with reference sign 32, a common shared level 2 cache 34 (L2-cache), and a common shared main memory 36.

[0065] The processors 32 are both working units and unit drivers. They generate requests to the L2-cache 34 and wait until the preceding request is complete before a new request is launched. In the schematic model depicted in Fig. 3, a statistical number generator is used to determine the time at which the next request to the L2-cache 34 is issued. Alternatively, in trace driven models, a trace reader working on an instruction trace would be used instead of the statistical number generator.

[0066] According to the preferred embodiment given in here, the time each processor 32 spends waiting for a request to complete μT is recorded in a table. Together with the processor's entry utilization, this wait time determines the actual load of each processor 32 at any time T. As detailed above, an accurate simulation of the CP utilization is key for a high quality performance model.

[0067] The control elements of the model are shown outside box 30.

[0068] The highest priority control element depicted with reference sign 38 is a module called service. When starting a simulation run, the service module 38 invokes an initialization routine 40 which provides all parameters describing the system configuration and the workload to be modeled. In particular, the initialization routine reads all the request rates and the desired target utilizations of the processors from input files associated with the workload to be modeled.

[0069] With regard to the target utilization μ, a first entry utilization χ'_1 is selected in a plausible way. From χ'_1 and from the request rates read-in before, the mean time $\tau=\tau_1$ between any

two requests is determined. Said mean time is referred to herein as interarrival time.

[0070] Next, the service module 38 starts as many instances of the processor modules 32, as CPs are to be simulated. As mentioned above, the processor modules 32 generate storage hierarchy requests based on the actually valid interarrival time of requests and record their duration.

[0071] During all simulation time, the service module 38 remains in a program loop, which invokes the statistics and the initialization routine in predetermined time intervals. The statistics routine corresponds to the utilization measurement device depicted with reference 20 in Fig. 2.

[0072] Upon each call, said statistics routine calculates the actual total utilization μ_n of the processors 32 from their entry utilization χ'_n and the table-stored request wait time as described above in the introductive theoretical section. The initialization routine in turn uses the inventive formula (6a, 6b) to calculate the entry utilization χ'_{n+1} for the next observation period from μ_n and μ. Finally, from the $\tau=\tau_{n+1}$ new value of the entry utilization, the mean interarrival time is re-calculated and used from now on by the processing units 32.

[0073] For improved clarity and as a supplemental inventive disclosure, the following section provides the pseudocode to the preferred embodiment of the invention illustrated by Fig. 3 and Fig. 4. For simplicity, the instantaneous entry utilization is denoted by y rather than χ':

```
Service                                        /* main controller */
{
Inits(virgin = TRUE)                   /* initial initialization*/
Generate CP-Drivers 0,...,m                    /* MP system */

Loop here for run_time
{
Advance Delta cycles                   /* observation period */
Stats(run_end = FALSE)                 /* calculate utilization */
Inits(virgin = FALSE)                  /* adjust entry util. */
}

Stats(run_end = TRUE)                  /* final run statistics */
}


Inits(virgin)                                  /* Initializations */
{
If virgin {                                        /* on run start */

Read input parms
specifically request rates r
and aimed-at utilization u
Make guess on entry_util y = y.1   /* initial entry util */

} Else {                                       /* each observ period */
                                                  /* use algorithm */
Calculate entry_util y = y.n = y(u.n,u)
                                                 /* u.n from stats */
}

Calculate interarrival time Tau = Tau(y,r) of requests
}
```

```
CP Driver                                   /* m = MP instances */
{
Loop here for run_time
{
Determine time to next request:                /* typically: */
t = Distribution(Tau)           /* Exponential distrib*/

Advance t cycles                            /* to next request */
Launch request
Wait until request_done

Record wait time w                          /* for stats */
}
}


Stats(run_end)                                  /* Statistics */
{
If (not run_end) {              /* each observ period */
                                     /* w recorded by CP */
Calculate actual utilization u.n = u(w,y.n)
                                /* y = entry util */
} Else {                             /* run complete */

Evaluate and report performance statistics
}
 }
```

[0074] Fig. 5 illustrates the efficiency and advantages of the inventive method by comparing its convergence behavior to prior art methods.

[0075] The first prior art method is the stepwise approach, the curve of which is depicted with reference sign 51: The instantaneous entry utilization gets adjusted by a fixed value d, that is $\chi'_{n+1}=\chi'_n+d$, which changes its sign and gets divided by 2 as soon as the utilization crosses the utilization aimed at: $d \rightarrow -d/2$. This is the approach with the poorest convergence among the alternatives shown.

[0076] Better convergence is achieved with further approaches depicted by the graphs with reference signs 52 and 53: Instead of the fixed value d used in method 51, they use the actual deviation from target utilization $d_n=(\mu_n-\mu)/\mu$ to adjust entry utilization. Two approaches are shown herein.

[0077] The straightforward approach $\chi'_{n+1}=\chi'_n/(1+d)$ is a second prior art method depicted by reference sign 52. A choice with noticeably better convergence exploits the instantaneous version of formula (11) which reads $\chi'_{n+1}=\chi'_n/(1+d_n(1+\rho'_n\ \chi'_n))$. This is an inventive embodiment of the general inventive formula (6a, 6b). Its graph is depicted with reference sign 53.

[0078] By far best performing is the approach which directly uses the inventive formula (6a, 6b). It is depicted with reference sign 50. Excellent convergence is reached at a time where prior art curves still show unacceptable high fluctuation.

[0079] The present invention can be realized in hardware, software, or a combination of hardware and software. A tool according to the present invention can be realized in a

centralized fashion in one computer system, or in a distributed fashion where different elements are spread across several interconnected computer systems. Any kind of computer system or other apparatus adapted for carrying out the methods described herein is suited. A typical combination of hardware and software could be a general purpose computer system with a computer program that, when being loaded and executed, controls the computer system such that it carries out the methods described herein.

[0080] The present invention can also be embedded in a computer program product, which comprises all the features enabling the implementation of the methods described herein, and which - when loaded in a computer system - is able to carry out these methods.

[0081] Computer program means or computer program in the present context mean any expression, in any language, code or notation, of a set of instructions intended to cause a system having an information processing capability to perform a particular function either directly or after either or both of the following:

a) conversion to another language, code or notation;

b) reproduction in a different material form.

[0082] While the preferred embodiment of the invention has been illustrated and described herein, it is to be understood that the invention is not limited to the precise construction herein disclosed, and the right is reserved to all changes and modifications coming within the scope of the invention as defined in the appended claims.